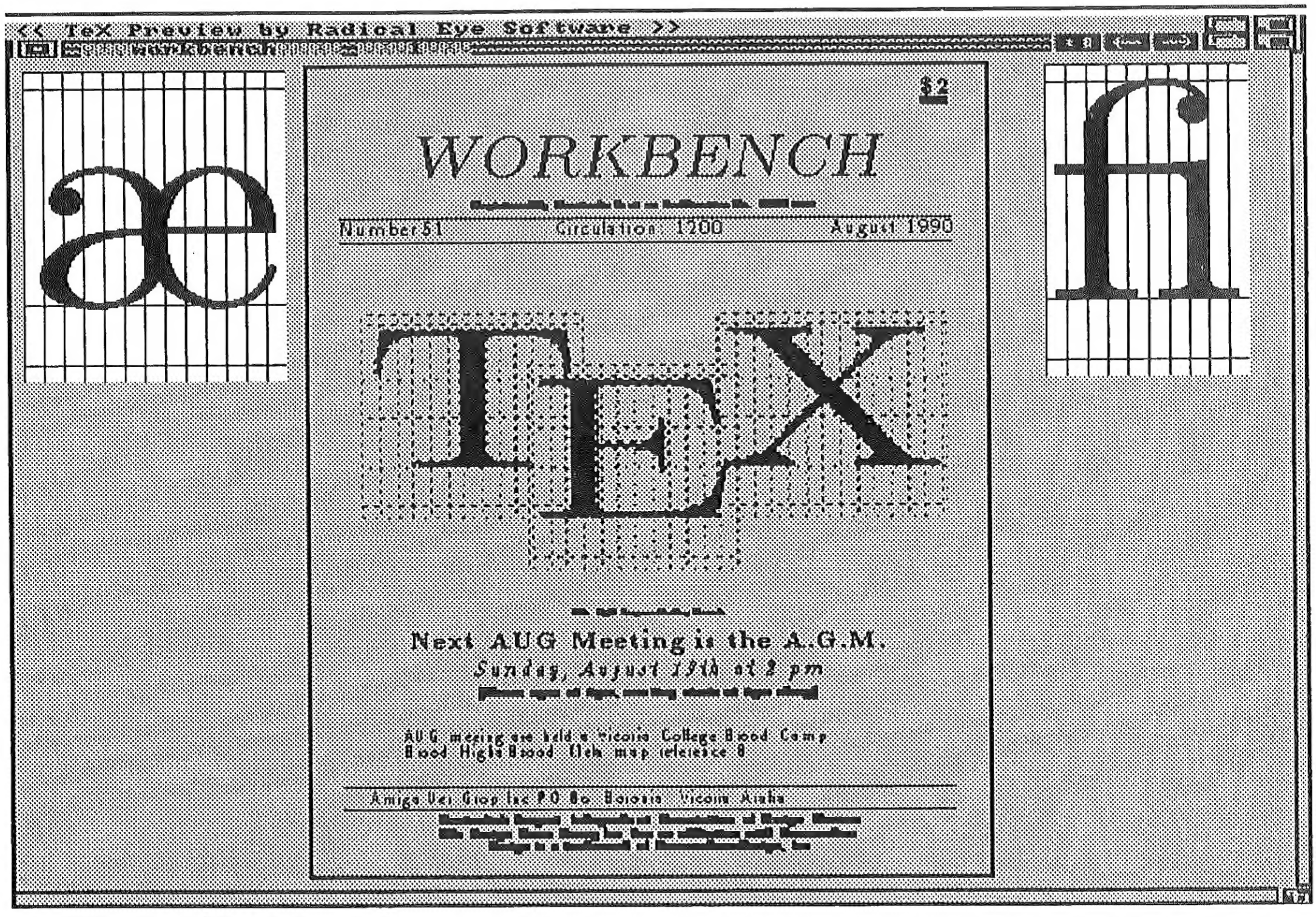
$3
RRP

AMIGA WORKBENCH

Registered by Australia Post — Publication No. VBG 7930

Number 51 Circulation: 1200 August 1990



The TEX Logo—Galley Proofs

Next AUG Meeting is the A.G.M.

Sunday, August 19th at 2 pm

(Doors open at 1pm, meeting starts at 2pm sharp)

AUG meetings are held at Victoria College Burwood Campus
Burwood Highway, Burwood – Melways map 61 reference B5.

Amiga Users Group Inc., PO Box 48, Boronia 3155 Victoria, Australia

AMIGA Users Group

Who Are We?

The Amiga Users Group is a not-for-profit association of people interested in the Amiga computer and related topics. With over 1000 members, we are the largest independent association of Amiga users in Australia.
We can now be reached by telephone—albeit an answering machine on:

563 9293

We do not condone software piracy nor do we permit any commercial software to be copied or exchanged at any of our meetings. Failure to comply with these policies will result in confiscation and erasure of the pirated copy. Remember, **software piracy is a crime.**

Club Meetings

Club meetings are held at 2 p.m. on the third Saturday of each month at Victoria College, Burwood Highway, Burwood. Details on how to get there are on the back cover of this newsletter. The dates of upcoming meetings are:

Sunday, August 19*th* at 2 p.m.—the A.G.M.

Sunday, September 16*th* at 2 p.m.

Sunday, October 21*st* at 2 p.m.

Meetings of Special Interest Group (SIGs) and Chapters can be found detailed separately in the newsletter.

Production Credits

This months newsletter was edited by Eric Salter and typeset with *AmigaTEX* in *Computer Modern Roman* 10 point. The camera-ready artwork was prepared from the TEX *dvi* file using a *dvi to PostScript* driver. The equipment used was an Amiga 2000 with 25 MHz GVP 68030 accelerator card and 4 Mbytes of 32-bit wide memory and *PostScript*-compatible laser printer.

Copyright and Reprint Privileges

Contributions

Articles, papers, letters, drawings, cartoons and comments are actively sought for publication in Amiga Workbench. All contributions published in the newsletter are rewarded with one free public domain disk copy per column or half page printed. Contributions should be submitted in machine-readable format, preferably on Amiga 880K format disk or via modem to AmigaLink I or II's newsletter up-load areas. Text should be in unformatted ASCII, diagrams should be in IFF format or be of camera-ready quality and photos should be glossy black and white prints. Please include your name and mail address for return of disks. Absolute deadline for copy is 23 days before the next meeting date. Contributions can be sent to: **The Editor, AUG, PO Box 48, Boronia, 3155.**

Membership and Subscriptions

Membership of the Amiga Users Group is available for an annual fee of $25. To become a member of AUG, fill in the membership form in this issue (or a photocopy of it), and send it with a cheque or money order for $25 made out to Amiga Users Group Inc.

Public Domain Software

Disks from our public domain library are available for $6 each including postage on quality 3.5″ AUG-supplied disks, or for $2 each on your own disks. The group currently holds over 300 volumes, mostly sourced from the USA, with more on the way each month. Details of latest releases are printed from time to time in this newsletter and a catalog disk is also available.

Membership Discounts

The Amiga Users Group negotiates discounts for its members on hardware, software and books.Currently, Technical Books in Swanston Street in the city offers AUG members 10% discount on computer-related books, as does McGills in Elizabeth street. Just show your membership card. Although we have no formal arrangements with other companies yet, most seem willing to offer a discount to AUG members. It always pays to ask!

Back Issues of Workbench

All back issues of Amiga Workbench are now available for $2 each including postage. Note that there may be delays while issues are reprinted. Back issues are also available at meetings.

Amiga Link I, II & III Bulletin Board Systems

The Amiga Users Group operates three bulletin board systems devoted to the Amiga, two using the OPUS message and conferencing software. AmigaLink I and II are available 24 hours a day. They may be accessed at V21 (300 bps), V22 (1200 bps), V23 (1200/75 bps) or V22 bis (2400 bps), using 8 data bits, 1 stop bit and no parity.
AmigaLink is part of a world-wide network of bulletin boards, and we participate in national and international Amiga conferences. AmigaLink has selected Public Domain software available for downloading and encourages the up-loading of useful public domain programs from its users. AmigaLink I (792–3918) is OzNet node number 8:830/324 and AmigaLink II (376–6385) is OzNet node number 7:1305/998.
The third BBS is also known as N.W.A.U.G. One and is run by the North-West regional SIG of AUG. It runs the *Skyline* protocol allowing the use of mouse and Amiga graphics to make things more Amiga-like. It is available on 376-7375.

Newsletter Advertising

The Amiga Users Group accepts commercial advertising in Amiga Workbench subject to the availability of space at these rates:

Quarter page $20
Half page $40
Full page $70
Double page spread $120

These rates are for full-size camera-ready copy only. We have no photographic facilities—typesetting facilities are available at extra cost. Absolute deadline for copy is 23 days before the meeting date. Send the copy and your cheque to: **The Editor, AUG, PO Box 48, Boronia, 3155.**

Amiga Users Group Committee

Co-ordinator	Neil Rutledge	578 5724	Caulfield
Ass. Co-ord	Chris Wearn	369 4302	Laverton
Meeting Chair	Arnold Robbins	808 0551	Box Hill Sth.
Secretary	Alan Garner	879 2683	Ringwood
Treasurer	Donna Heenan	543 7415	Clayton
Membership	John Hampson	N/A	Cheltenham
Purchasing	Bohdan Ferens	792 1138	Dandenong
Book Library	Ross Johnson	824 7026	Malvern
Disk Library	Michael Lamb	752 2130	Ferntree Gully
Editor	Con Kolivas	484 1339	Thornbury
Committee	Peter Jetson	752 2053	Upper F'tree Gully
	Rod King	(059) 62 5805	Healsville
	Chris Tremelling	557 1349	Bentleigh

NWAUG Committee—see inside

About This Issue

by Eric Salter

WELCOME to the first edition of *Workbench* that has been typeset entirely by AmigaTEX. I have played with AmigaTEX for about 18 months now and I have been dying to get my teeth into something like this journal that has some difficult formatting requirements.

AmigaTEX is a typesetting program for the Amiga but many other computers can run TEX including Macs and IBMs but no implementation has the same integration and power as that for the Amiga; written by Tomas Rokicki—well known for his many public domain utilities, AmigaTEX allows the Amiga to become a powerful typesetting tool, with control over the finest details of typographic design.

Just a word about the pronunciation of 'TEX'—it is derived from the Greek word '$\tau\epsilon\chi\nu\eta$' meaning *art*, so it reads as *tech* with much spitting on the '*ch*.'

TEX is a text compiler, just like your 'C' or Pascal compiler. Instead of lines of code, TEX's input is the lines of your document interspersed with directives to TEX to format those lines in a particular way—i.e., TEX is a declarative language a little like Prolog.

Plain TEX is very boring; It is possible to just let TEX loose on a piece of text having included an '`\end`' command at the end of the document; You'll get a nicely typeset slab using the default format—quite uninteresting and useless. On the other hand, if you want TEX to typeset something like *Workbench*, then you've got a lot of work to do. The macros that I wrote to make TEX format these articles in the *Workbench* tradition amount to some 12K (345 lines) of inscrutable TEXese.

While it is hard to make TEX do weird and wonderful things to text, it is equally hard to make TEX let you err typographically. You'll see other journals put together with desk-top publishing packages where the feel of the printed page changes from page to page and each page is filled with more font changes than text. TEX makes you consistent—you're page will look the same at the beginning of a book as at the end. Spacing of characters and lines are calculated down to an accuracy approaching 100 wavelengths of visible light. Once the formatting commands are written, the text of the document is molded and lovingly massaged until TEX thinks it looks right—a sort of printers *imprimatur*.

Anyway, getting down to the nitty gritty; For those who are interested in typography, this journal has been set in a font called *Computer Modern Roman* at 10 point size with leading of 12 points. TEX's tolerance of bad lines has not been adjusted for the narrower column width nor has its penalty for hypenating words—which just goes to show you don't have to compromise!

A few people in the AUG are using TEX and it would be a great idea if we could form a special interest group to educate ourselves and learn how to tame this beast. TEX has a very steep learning curve but in the end, the results are worth it. If you would like to know more about typeography or even desk-top publishing, please contact me on 853–9117.

Finally, I hope you enjoy *Workbench* in this format but I hope those people who are more skilled at typography and TEX than myself, are not too critical of my mistakes.

Game Reviews

Playing Empire

Public Domain Game (Fish #329)

by David Peel

WITH our Amigas we get a lot spoilt with the sort of expectations we have of games. Even the public domain games use all the features of the Amiga and they are exciting and fun. So what do you do when you see a game like *Empire* you expect graphic screens the ability to use mouse etc. Well the joke's on you buddy. Some of you may remember back in the old days, when on mini and main frame computers those earnest users often whiled away the odd one or two hours playing *Adventure*. This led to games like *Zork* where you basically played word games and tried to work out where in the maze (hell?) you were. *Empire* is a bit like that and a bit like a sort of Dungeons and Dragons. I say a bit because it is really only a bit like the games I've mentioned.

With Empire a certain size world is created, and depending on your choice of size you can end up with two countries or 128. This means that you have two players or up to 256 players.

The first problem I found when I decided to play it was how on earth to get started. I don't have a modem so the local keyboard game was my natural choice. I made up the three disks as required and named them as required. Then I transferred the appropriate files to the diskettes. It took me a while to be able to figure out how to actually create a world. You have to read the *readme* file literally so that when it asks for a creation password such as "*Godpassword*" and "*creationpassword*" that is what you have to give it. you then get to name a country and to log in to that country you have your own secret password. When you play the game you start by logging into the game or bulletin board on which the game is and then you are told that you have been assigned two sectors which are a sanctuary. If you type the `map#` command you see these two sectors on a grid 2×1. they would be respectively $0,0$ and $1,0$. When you move from these centres they become your capital city generating BTU's which are *bureaucratic time units* and these are the means by which you are able to move and live. You move out from the capital by moving civilians or troops using a command like:

```
Move C from Sector 0,0 to 0,1
```

You can move only as far as the limits of the current map, but as you move out this grows. I find it easier to allow the programme to prompt me for moves as I can never remember the exact syntax.

In Play—it is fun but it takes a lot of fiddling and also strategy. It is a bit gruesome when you say move 40 military personnel into a sector and you are told they have drowned because it's a sea sector. The game updates as you play (this is apparently done by God, who owns the world) so you might find plagues earthquakes or bumper harvests. I

am still trying to work out how to increase production and build ships and trade with others build highways etc, etc.

I would be happy to talk to anyone who thought they might like to play it and doesn't want to re-invent the wheel. I can at the very least avoid the pitfalls at the early stages and get them up to the level of working out the strategic aspects of the game. There is of course every chance that they might beat me at it! There are some aspects of the game which I would like to see changed. I would for instance, like to see some non military production capability as well as some diplomatic facility. There is the implicit assumption that all growth and expansion must necessarily lead to war and destruction. Other options such as peaceful mergers and co-operation do not seem to be allowed for or encouraged.

If some whiz-kid could make it make pikkies of what is happening that would be super but I be reasonably happy if some showed me how to pipe the output straight into my text editor so that I could capture maps. My present scheme of using *snipit* to cut and paste to *AZ* the editor I use shows up a funny quirk of Snipit when you cut more than one line. This quirk is that it moves the next line over about half a page width which is a bit of a pest. Other than that I think it is a great game so good in fact that I think I'll have a look at the update on Fish #357.

Editor's Note: *The interested reader may like to read another article on* Empire *by "Saint Nikolai" in the December 1989 Workbench.*

Falcon

Flight and Mission Simulator

by A. R. Fisk

FALCON is a flight and mission simulator for the General Dynamics F-16 Fighter-Bomber and is produced by *Spectrum Holobyte*. The software comes on two unprotected disks, a code wheel and a *very* comprehensive manual as well as a quick reference sheet. After going for a cuppa while the game loads and applying a bit of fuzzy logic to matching the given symbols on the code wheel (which are sometimes a little difficult to figure out), one is provided with a duty roster list on which the names of all the budding Biggles' are to be recorded.

Such formalities over, the player is presented with a selection of twelve missions which range from the *Milk Run* which is a simple 'circuit and bumps' with a bit of ground target practice thrown in through to *Grand Slam*, where you are required to down at least five enemy aircraft. The missions provide a wide range of scenarios and are flexible in permitting secondary targets such as enemy aircraft and SAM sites.

The mission you choose to undertake will decide your weapons load which may be selected from the Armament Screen. Possible selections include AIM-9L and 9J Sidewinders, AGM-65 Mavericks, 2000 lb. bombs, Durandal anti-runway bombs, ECM pods and external fuel tanks. A cannon with 5000 rounds of ammo is included. While missing out on the choicer items of military hardware such as anti-radar missiles and AMRAAM's this selection is still reasonably varied. Your choice of ironmongery will be assisted (and criticised!) by 'Sarge' the Ground Crew Chief. Certain items may not be available, especially at higher ranks, so be prepared to make do with what you can get.

The difficulty of each mission varies according to the rank you choose. At Flight Lieutenant, the lowest of the five levels, your *Falcon* will have a souped-up engine with unlimited fuel and weaponry plus a useful ability to bounce off or fly through anything. These advantages are removed as rank increases. Flying characteristics become increasingly strict. Munitions may be unavailable and their weight will affect the plane's agility. Enemy MIG-21s start emitting flares to evade your missiles and will shoot back with increasing accuracy. High-G manoeuvres may result in a blackout. Worse still, you will come under fire from a variety of SAMs from both fixed and mobile installations.

One further level of difficulty is attained by varying the number of enemy fighters (up to three) trying to move in on your six at any given time.

In addition to the formal mission scenarios there is also a practice *dogfight mode* in which the target aircraft attempts to evade you by using a variety of textbook manouevre, each of which are described in the manual. For those Colonels who find those MiGs a touch on the wimpish side there is one further enhancement: a selection which makes your opponent an Ace pilot.

For some real fun, you can combat an opponent on a second machine via the serial port. The second machine may be any of those for which *Falcon* has been released; so it may be another Amiga, or Atari ST, or Mac...but not, it appears, the humble PC!

If you want to analyse your combat style, and have 1Mb available, there is a 'black box' option allows you to playback the flight paths of all aircraft. This is a very useful feature if you are having trouble working out why you keep getting shot down. This feature is the only one requiring memory expansion by the way. Other than this *Falcon* should work quite satisfactorily on 512K.

Although *Falcon* places a heavy emphasis on performing combat missions, there is a fair amount of challenge to be found in just learning to fly the beast! The novice is strongly advised to forget the "shooty bang-bangs" to begin with and to concentrate on becoming a decent pilot.

At the start of each mission you will find yourself on the main North runway cleared for takeoff. Air traffic controllers may beg to differ here but now is an excellent time to hop out for another coffee, put your feet up, and read the instructions (whose size and comprehensiveness are probably a more effective hedge against piracy than the code wheel!).

The cockpit has an initially bewildering array of instruments both on the console and on the Head Up Display. Still more are to be found on your side views although these are generally backups and you shouldn't need to look at these too often. Each instrument is clearly explained in the manual and the layout is supposedly quite true to life although one omission that has been noted elsewhere is the lack of a vertical air speed indicator. On the other hand, if the altimeter is madly whizzing round in an anticlockwise direction then one can safely deduce that one is in a dive.

Learning how to use all those gizmos is the next step. The manual does this by taking you on a step by step basis through the Milk Run mission starting from the correct take-off procedure through to manoeuvring the plane to approach the ground targets (any of three buildings a few miles to the East of the airfield), making your attack run and finally, getting yourself back down in one piece.

Your first missions can be carried out with some detours to get familiar with the plane's handling characteristics and to admire the scenery. You have a choice of 4 cockpit views (front , back and sides) as well as 3 outside views of the aircraft (satellite, birdseye, and control tower). The satellite and birdseye views have a zoom control and the satellite view also has a 360 degree pan control. While all this may sound comprehensive I have found the cockpit views to be a little limited; it is virtually impossible to maintain visual contact with another fighter if it moves out of your front view unless you are level (not advised in a dogfight!). The satellite view comes in useful here but is hardly realistic! Besides, you can only pan the view in one direction. I feel that a fifth view allowing the pilot to look straight up would have been a simple matter to include and would have improved the supposedly all-round visibility immensely. That said, I know of no simulator that does include this feature (not that I know many!).

Getting back to the scenery. Well, there's your home base, with two runways, a control tower and a hangar. (A quick point, don't try taxiing into the hangar at higher ranks, you crash!). Further away you can see a modest selection of mountains (again, you can crash into these at higher ranks), lakes and rivers. A number of other artificial landmarks, such as roads, bridges, fields, SAM sites, enemy runways are marked on the map. What aren't marked are the telegraph lines running alongside the roads (complete with insulators on the poles), or the farm buildings near the fields. This marvelous attention to detail presented to someone who is normally passing it all at several hundred km/h at an altitude of several thousand feet puts one in mind of those cathedrals where the rafters far from public attention are covered in intricate carvings. I discovered another nice touch when I was looking out for bogies and saw a moving white speck which wasn't registering as a threat. It turned out to be another *Falcon*!! How does all this look? Pretty damn good and motion is very smooth, unless you get jumped by two or three MiGs, in which case sightseeing is probably the last thing on your mind. At such times the action can be speeded up by removing the scenery.

Although it can provide a very satisfactory flying simulation, *Falcon* tends to place its emphasis on combat situations and refinements such as wind, turbulence and clouds are not included. Certainly, all these factors can play a part in real combat, but then, the name of the game is *Falcon*, not *Cessna*. One thing about the combat that does annoy me is that while MiGs can be left out of a pure flying mission, SAMs may not. This isn't important at lower ranks as SAMs are only effective at Major rank and above. Unfortunately, this is the same point at which the 'super' falcon performance characteristics cut out. You have to learn to fly a much trickier aircraft without being able to remove the bother of being shot at, unless you stick to your home turf.

Minor grumbles notwithstanding, *Spectrum Holobyte* have succeeded in making *Falcon* a highly playable and absorbing game while at the same time retaining a high degree of realism in the aircraft's handling capabilities. It sells for $59.95 and is highly recommended.

TV Sports: Football

to excite the competitive spirit

by "Night Stalker"

THIS high quality simulator generates challenging scenarios guaranteed to excite the competitive spirit in any sports fan! Gridiron is a growing game in this country and *TV Sports* offers a window into a sport which can really get into the blood once one understands what's going on.

TVS is a two-disk package accompanied by documentation which provides sufficient info to acquaint one with the fundamentals of the sport. The program disk has a form of copy protection which has defied all my attempts (via countless copy progs.) to create a working back-up; This unfortunately prevents transfer to a hard-disk.

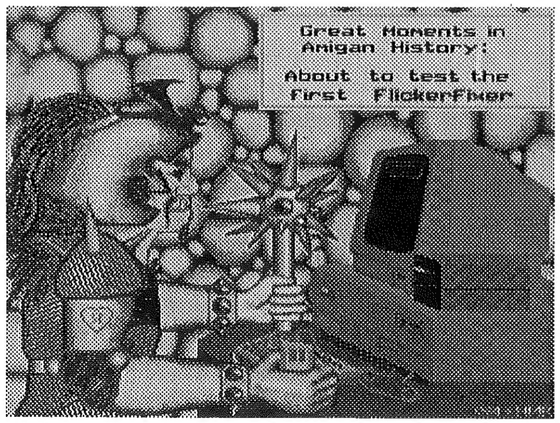
Features include practice mode, exhibition games, one or two players as either opponents or team mates vs. computer opponent, 16 round league season playoffs and Superbowl final. The league has 28 teams each with the option of human or computer control which allows upto 28 players to compete. Unlike the bulk of simulators, this one provides excellent computer opponents, so don't despair if you have no suitable friends at hand.

Good quality graphics with stereo sound effects combine to create plenty of atmosphere including a vocal home crowd, well-endowed cheer leaders, a variety of half-time shows complete with brass band, novelties and verbose commentator. The average game takes 90 minutes and the serious player will use every second of that time to maximise performance, since there is no *pause* facility and it costs valuable time-outs to have a breather.

A very user-friendly feature allows varying degrees of player participation in a game. This gives the learner freedom to concentrate on individual aspects rather than attempt to fully control the team prior to acquiring the skills to do so. First priority ought to be the offense, which entails coaching, calling effective sequences of plays and successfully executing the plays on-field as the Quarterback. Your defense will run itself quite well while you are working at putting scores on the board and learning how to beat defenses—which will make it all the easier come the time when you are ready to manage your own. How will you know when you're ready? Don't worry, you'll know!

If this intro has whet your appetite, then scrape the $75 (approx) together and treat yourself to a package of superior quality, which I can tell you has proven to be my most-played simulator and will be so for some years to come. Oh, by the way, if our editor is willing to deal with the workload [*Ed: Not if I have to type it in by hand—I prefer machine-readable copy*], then next month I'll discuss the selection, designing and testing of your team(s)

and other necessary pre-season issues, then follow-up with the in-season topics including strategy, tactics, game planning and stats.



Amiga Users Group

Annual General Meeting

To be held at

Victoria College, Burwood Campus

on

Sunday August 19th

at

2 p.m.

At the Annual General Meeting, the Officers of the Association and members of the committee will be elected. Reports from the Co-ordinator and Treasurer as well as other members of the committee will be heard.

Co-ordinator's Comments

HERE is a description of all the positions to help guide potential new committee members:

Co-ordinator (President)–Has the responsibility of overall co-ordination and control of the groups activities. This position offers an opportunity for any member of AUG to become strongly involved and help set the future direction of the group.

Assistant Co-ordinator (Vice-President)—This position exists primarily as a support to the co-ordinator but has in the past taken on the specific responsibility of co-ordinating the SIG groups and special activities undertaken by AUG and its members.

Meeting Chairman—Is in general required to organize the events at the AUG monthly meetings and with assistance from other committee members arrange demonstrations and displays of items considered to be of interest to general group members.

Secretary—Performs the usual functions expected of the secretary of any club or user group such as general correspondence, collection and distribution of committee meeting minutes and booking of meeting rooms.

Treasurer—Again a fairly traditional position involving the management of the daily finances of the group e.g., collecting and banking of funds received from P.D. disk sales, membership payments etc.

Membership Officer—Responsible for managing membership of the group by collection of dues, updating of database and issuing of renewals.

In addition to the above formal positions there are four general committee positions to cover the additional requirements of helping to run a group the size of AUG.

The positions of Newsletter Editor, Software Librarian, Book Librarian and Purchasing Officer can be held by ordinary members of the group and are usually selected by committee members.

This summary is offered as a guide only and is not to be considered a formal position description, the rules of the association define the specific requirements in more detail. The group would also fail to operate without the support and work put in by the co-ordinators of the Special Interest Groups. The committee meets once a month to discuss group business and other duties can normally be handled by phone contact or at the main monthly meeting. To those who have a desire for the club to move in a specific direction or would like to put some time towards the clubs future come along to the August meeting and nominate for a position.

—*Neil Rutledge* (Co-ordinator)

Learning CLI

What Does That Mean? #1

Pure Bit Not Set

by Nikolai

YOU may have seen this message when you were booting with a copy of a *Workbench* 1.3 disk. It's nothing to worry about (usually), but to explain why that message appears requires discussion of two very CLI-type things:

1. Protection Bits

Apart from all the other stuff that's included in a file, AmigaDOS also keeps track of seven *Protection Bits* that determine what we're allowed to do to the file, among other things. These bits (think of them as Flags) are known as `Read`, `Write`, `Execute`, `Delete`, `Script`, `Pure` and `Archive`. The current revision of AmigaDOS only takes notice of the last five. If the `Execute` bit ISN'T set on a program file,

then you won't be able to run it—AmigaDOS will give you a message:

```
Execute permission not set
Unable to load name:  file is not an object module
```

If the `Delete` bit isn't set, you won't be able to delete it. This affects Workbench users as well; if the `Delete` bit isn't set on a file's Icon file, then you won't be able to move that Icon around and have the Workbench remember its new position: this is because when you do a '*snapshot*' of an Icon to save its new position, the Workbench copies the Icon and deletes the old one (I ran into this while trying to clean up *Digiview* 3.0—all of the files are protected in this way). You can see how a file is protected from the Workbench by selecting its icon and then selecting '*Info*' from the Workbench menu. The protection bits are displayed on the top right.

Well, `Read`, `Write`, `Execute` and `Delete` are fairly self-explanatory; the `Archive` flag is used for letting programs know if a file has been modified since it was last saved on a backup disk somewhere. For example, if you paint a picture in *DeluxePaint*, and then back it up to another floppy disk, the `Archive` flag should be set (by the program you use to back it up). Then, if you use *DeluxePaint* on that file again, the `Archive` flag should have been reset (AmigaDOS doesn't do this automatically—it's up to the program). Sounds like a lot of bother, doesn't it? The convenience comes when you want to back up all the files that have been modified since you last did a backup—you can just say 'copy anything that hasn't got the `Archive` flag set.'

The `Script` file lets AmigaDOS treat a straight text file like a compiled program file; so if we have a file called '*Fred*' full of AmigaDOS commands and '*Fred*' has the `Script` flag set, then you can just type '*Fred*' at the CLI to execute it, instead of '`Run c:Execute Fred`.'

The one you've all been waiting for—the `Pure` flag. This one requires discussion of the other CLI thing I referred to previously.

2. Resident Commands

One way of speeding up AmigaDOS is to copy the `C:` directory to `RAM:` and tell AmigaDOS that if it want a `C:` command, to look in `RAM:` for it (with the `ASSIGN` command). A quicker way of doing this is to add the command to a special list called the *Resident List*. Resident commands are kept in memory, saving AmigaDOS the trouble of loading them each time you execute them. Before you can make a command memory-resident, it must have been written in a certain way, so that more than one process can run it at once. If a program is written in this way, it is described as *Pure*, and usually has the `Pure` flag set. You can add non-pure programs to the Resident list, but if you do, you're usually asking for a visit from the Guru when you try to run them more than once (that's what multitasking is all about, folks).

Now, if you add a program to the resident list that doesn't have the `Pure` bit set, you will get the message '`Pure bit not set`' (unless you specifically say 'I don't care if it's pure or not, `ADD` it already'); and, if you've made a copy of the Workbench Disk in such a way that the protection flags weren't copied, all of the pure commands in the `C:` directory (and there are quite a few of them) will still be pure, but the Resident command won't know that. So, what can you do?

Some Workbench-based utilities (like *DirUtil*) will let you alter the protection bits, but if you aren't sure which programs are pure and which aren't, then it's best to simply copy them from the original Workbench disk, making sure that the protection bits are copied as well. Or, you can find the part of the `S:Startup-sequence` that makes the commands resident, and make sure it looks like this:

```
resident c:execute pure

    (or)

resident CLI L:Shell-Seg SYSTEM pure add
```

`resident, CLI, L:Shell-Seg`
- the names of the commands being made resident

`SYSTEM`
- this means 'Add it to the `SYSTEM` list' which means we can't UN-resident it again later

`pure add`
- this means 'Add it even if it isn't pure'

and as always, remember the thirteenth commandment:

> *'Thou shalt not stuff around with thy original Workbench Disk—use a copy.'*

(and if you're going to make a copy, use *DiskCopy* or the Workbench to do it—that way, all of the protection bits get copied.)

p.s. There is, in fact, an eighth protection bit—a `hidden` flag, but as far as i can tell, it doesn't do anything; it's probably there for future revisions of AmigaDOS.

Programmer's Corner

Machine Code and BASIC

by John Elston

BASIC has many virtues one of which is the speed of development and debugging of an application, however in terms of execution speed BASIC tends to be somewhat on the slow side. A BASIC compiler will help but in some cases even this will not provide a sufficient improvement. On the other hand machine code executes very quickly but is time consuming to develop and debug programmes. The obvious solution is to combine the two—develop the main programme using BASIC and use machine code subroutines for the speed critical sections.

The Amiga BASIC manual is not much help as the single example given does not appear to work. The programme below is based on the one given in the manual with one advantage—it works (I hope!). Basically (pun intended) all the subroutine does is to change the value of *Y%* to that of *X%* by passing *X%* and the address of *Y%* to the subroutine.

The manual states that parameters are passed by value using 'C' language conventions. This means that the return address is stored at a location pointed to by the frame pointer (A6) followed by the integers in the parameter list. To get the address of the first parameter we add 8 to the A6 register, the address of the second parameter is A6 + 12, the third A6 + 16 etc. (note numbers are decimal).

My routine saves the registers, moves *X%* into A0, the address of *Y%* into A1, then moves *X%* (A0) into the location pointed to by A1 (the address of *Y%*), recovers the registers from the stack and returns to BASIC.

Be aware that the Guru is easily invoked using these methods, particularly in a multitasking environment perhaps a reason for the paucity of information on this subject.

```
' Created 16th June 1990 by J. S. Elston
' as an example of calling a machine code routine
' from BASIC. Based on the example given in the
' AmigaBASIC manual page 8-154.

DIM Code%(60)

WHILE Byte <> &H4E75
   READ Byte
   Code%(Index%) = Byte
   Index% = Index% + 1
WEND

X%= 10 : Y% = 0

SetYtoX= VARPTR(Code%(0))
CALL SetYtoX(X%, VARPTR(Y%))

PRINT "Y%= ";Y%

END

' preserve registers
'   link a6,#0
DATA &h4E56, &h0000
'   move.l 8(a6),a0    ;move X% into a0
DATA &h206E, &h0008
'   move.l 12(a6),a1   ;move the address of Y% into A1
DATA &h226E, &h000C
'   move.w a0,(a1)     ;move X% (A0) into the address
'                      pointed to by a1
DATA &h3288
' Restore registers
'   unlk a6
DATA &h4E5E
' now return from whence we came
'   rts
DATA &h4E75
```

Changing Mouse Sprites

A 'sleep' pointer sprite in 'C'

by John Morton

ONE day I was messing about with sprites on the Amiga and decided it was time to do something practical. So I came up with this small but invaluable module—*Sleep*. What *Sleep* does is change the mouse pointer to the 'snooze' cloud, so you can show your user that you are busy doing something else. When you are ready to begin accepting user input again, you can change your 'snooze' cloud back to the normal pointer. It is all quite simple really—with a call to *Sleep* with the FLAG set to TRUE, means "send the pointer to sleep"; a call with the FLAG set to FALSE, means "wake my pointer up."

You know, like me, lots of Amiga programmers have written programs where this sort of call would add that professional touch, instilling confidence in your program and its intent.

This was written under *Aztec* 'C' 3.2, but I'm sure it will work just as well with your favorite compiler. Remember that it is only a single module and that you will need to have opened `intuition.library` and a window first before you can call it. I hope you will find it useful.

```
/*.....................................................*/
/*  Sleep Function—JCM @ Melbourne FrameWorks  */
/*          TAKA @ Melbourne FrameWorks        */
/*.....................................................*/
/*.....Data for Sleep Sprite.....*/
USHORT SleepData[ ] = {
  0x0000, 0x0000,
  0x03e0, 0x0000,
  0x0ff8, 0x0000,
  0x1ffc, 0x0000,
  0x7ffe, 0x0000,
  0x7fff, 0x0f00,
  0xffff, 0x0200,
  0xffff, 0x0400,
  0xffff, 0x0f00,
  0x7ffe, 0x0078,
  0x3fff, 0x0010,
  0x7fff, 0x0020,
  0x3ffe, 0x0078,
  0x1ffe, 0x0000,
  0x03f8, 0x0000,
  0x00f0, 0x0000,
  0x0fc0, 0x0000,
  0x1ff0, 0x0000,
  0x1f7c, 0x0000,
  0x0e20, 0x0000,
  0x00e0, 0x0000,
  0x01f8, 0x0000,
  0x03fc, 0x0000,
  0x01b8, 0x0000,
  0x0000, 0x0000,
  0x0000, 0x0000,
  0x0000, 0x0000
};
```



```
/*.....Data for Wake Sprite.....*/

USHORT WakeData[ ] = {
  0x0000, 0x0000,
  0xfc00, 0xfc00,
  0xb800, 0xc000,
  0xf800, 0x8800,
  0xfc00, 0x8400,
  0xfe00, 0xa200,
  0x9f00, 0x9100,
  0x0f80, 0x0880,
  0x07c0, 0x0440,
  0x0380, 0x0200,
  0x0100, 0x0100,
  0x0000, 0x0000,
  0x0000, 0x0000
};

/*
Module:   SLEEP—Change the mouse pointer sprite.
Date:     30/09/89
Author:   JCM
Purpose:  Changes the mouse pointer to a predefined
          sprite.
Usage:    Sleep( ptr_to_window, (WakeUp = FALSE,
          Sleep = TRUE))
          so to put your mouse pointer to sleep:
               Sleep( window, TRUE );
          now to wake the pointer up again:
               Sleep( window, FALSE );
Comments: Sleep will remove all messages from the
          Window's IDCMP that may be queued while
          the pointer was snoozing.
          There seems no point in actioning any
          invalid messages.
*/
/*  Previously opened intuition.library and a window  */

extern struct IntuitionBase *IntuitionBase ;

/*.....................................................*/

void Sleep( MyWindow, FLAG )
struct Window *MyWindow ;
BOOL FLAG ;

{
   struct IntuiMessage *message ;

   if ( FLAG )
      SetPointer( MyWindow, &SleepData[0],
                  24L, 16L, -8L, -8L ) ;
   else {
      while ( message = ( struct IntuiMessage * )
                  GetMsg( MyWindow→UserPort ) )
         ReplyMsg( message ) ;
      SetPointer( MyWindow, &WakeData[0],
                  10L, 10L, -2L, -2L ) ;
   }
}
```

AmigaDOS Resident Libraries

or "The hunk that time forgot."

by Eric Salter

THERE is more to AmigaDOS than meets the eye. Not only is it one of the most fascinating libraries in the Amiga's operating system (it is written in BCPL—a language foreign to most of us), it has some very useful features that are barely mentioned in the system documentation. This paper deals with one such "undocumented" feature—The AmigaDOS loader resident library support.

Libraries, libraries and "libraries"

What is a library? Well, the answer depends on what area of the system software you're dealing with. Exec-style libraries are well known to us—their "library node" structure is linked into Exec's library list during their initialisation process. We open these libraries by a call to Exec's *OpenLibrary*() function and their services are accessed by jumping subroutine to a negative offset from the library base address returned to us from *OpenLibrary*(). The offset for a particular library service is unchanging from one release of system software to the next.

Scanned or link-time libraries are familiar to those of us who write their own code. They are the libraries, such as `amiga.lib` and `lc.lib`, from which unresolved references to routines or other symbolic data in our 'C,' assembler or other high-level language program, are finally resolved by incorporating modules of code from these libraries to satisfy the linker.

The third class of library in the Amiga system is little known—or rather, known in a more traditional form. This is the AmigaDOS resident library which has been poorly documented, rarely mentioned, and has languished in obscurity from the year dot, but which offers some features that deserve to be explored further by the Amiga community.

It is known that Commodore are thinking of removing the code support for load-time dynamic linking of resident libraries from the DOS loader. I have written this paper to document this feature fully and demonstrate that it is a useful adjunct to the programmers toolbox in addition to the excellent features of this remarkable machine.

Resident Library Load-time Linking

The DOS loader provides for the automatic opening of system libraries by the *OpenLibrary*() call, and relocation of references to these libraries at load-time. The current practice with high-level language and assembler code programmers is to have some well-debugged standard startup code that opens the Exec, DOS and Intuition libraries and places their base address in static global storage. With the DOS loader, it is possible for a program to specify to the loader what libraries it is dependent upon for operation. The loader will open these libraries for the program and then relocate references to these libraries during the load. For example, the assembly code to call the DOS library function *Read*() could be written:

```
TheRead: movea.l     DOSBase,A6    ;load DOS base
         jsr         _LVORead(A6)  ;call the routine
         :
         rest of code
```

without ever having explicitly opened the DOS library to obtain *DOSBase*. Alternatively, it is possible to have the loader fix up references so that only:

```
jsr Read
```

is required to call the DOS *Read*() function, once again without having opened the library formally.
There are some major advantages to using this technique:

- Code is only loaded if the libraries are present
- Libraries are opened and closed automatically
- Less code overhead in the source and load files
- Slightly faster branches—absolute vs indexed
- Possibility of RUN-time linking of different programs, not just modules within a single program

There are however, some disadvantages:

- Blink doesn't support resident libraries—only Alink at the moment
- Commodore may remove support in the loader!—a major setback
- No opportunity to tell the user what library is missing (only a terse `ERROR 122: invalid resident library during load`)
- Version of library opened is "don't care"
- Slightly more complex linking phase

How is all this possible? In the "*AmigaDOS Technical Reference Manual*," terse reference is given to the process. We read:

> "Load files are also known as 'libraries.'...You can reference resident libraries through external references; the definitions are in a hunk containing no code, just a list of resident library definitions. Usually, to produce these hunks, you assemble a file containing nothing but absolute external definitions and then pass it through a special software tool to convert the absolute definitions to resident library definitions. The linker uses the hunk name as the name of the resident library, and it passes this through into the load file so that the loader can open the resident library before use."

and this is about all the information we get. What is this special "software tool" and where do you get it? Well, I wasn't able to find it (I didn't look very hard), but read on for a solution. [*Ed: The program* mkres *is not listed due to space restrictions. If interested in the code, contact the author*]

Creating Resident Library Definitions

To create a definition for a resident library, we first create a module of external references. Here is a simple definition for a few routines of the DOS library:

```
        SECTION dos.library,DATA

* create the offsets

DOSBase EQU   0
Open    EQU   -30
Close   EQU   Open-6
Read    EQU   Close-6
Write   EQU   Read-6
Input   EQU   Write-6
Output  EQU   Input-6

* make definitions external

        XDEF  DOSBase
        XDEF  Open
        XDEF  Close
        XDEF  Read
        XDEF  Write
        XDEF  Input
        XDEF  Output

        END
```

When assembled, the object file contains a series of absolute definitions for the symbols we have defined. We then pass this file through our data swabber to convert the absolute definitions into resident library definitions. The above code generates the following object code:

```
0000:  000003E7 00000000 000003E8 00000003 ...?.......?....
0010:  646F732E 6C696272 61727900 000003EA dos.library....?
0020:  00000000 000003EF 02000002 4F757470 .......?....Outp
0030:  75740000 FFFFFFC4 02000002 496E7075 ut..????....Inpu
0040:  74000000 FFFFFFCA 02000002 57726974 t...????....Writ
0050:  65000000 FFFFFFD0 02000001 52656164 e...????....Read
0060:  FFFFFFD6 02000002 436C6F73 65000000 ????....Close...
0070:  FFFFFFDC 02000001 4F70656E FFFFFFE2 ????....Open????
0080:  02000002 444F5342 61736500 00000000 ....DOSBase.....
0090:  00000003 000003F2 .......?
```

This code contains:

- `hunk_unit`—with no name field
- `hunk_name`—`dos.library`
- `hunk_data`—with no data
- `hunk_ext`— with 7 absolute external definitions:

Output	value `FFFFFFC4`
Input	value `FFFFFFCA`
Write	value `FFFFFFD0`
Read	value `FFFFFFD6`
Close	value `FFFFFFDC`
Open	value `FFFFFFE2`
DOSBase	value `0`

- `hunk_end`

After passing this object code file through *mkres*—the data swabber, the code looks like:

```
0000:  000003E7 00000000 000003E8 00000003 ...?.......?....
0010:  646F732E 6C696272 61727900 000003E6 dos.library....?
0020:  000003EF 03000002 4F757470 75740000 ...?....Output..
0030:  FFFFFFC4 03000002 496E7075 74000000 ????....Input...
0040:  FFFFFFCA 03000002 57726974 65000000 ????....Write...
0050:  FFFFFFD0 03000001 52656164 FFFFFFD6 ????....Read????
0060:  03000002 436C6F73 65000000 FFFFFFDC ....Close...????
0070:  03000001 4F70656E FFFFFFE2 03000002 ....Open????....
0080:  444F5342 61736500 00000000 00000000 DOSBase.........
0090:  000003F2 ...?
```

which contains:

- `hunk_unit`—with no name field
- `hunk_name`—`dos.library`
- `hunk_resident`—a previously undocumented hunk
- `hunk_ext`—with 7 resident library definitions:

Output	offset `0x3C`
Input	offset `0x36`
Write	offset `0x30`
Read	offset `0x2A`
Close	offset `0x24`
Open	offset `0x1E`
DOSBase	offset `0`

- `hunk_end`

We will now link this hunk to a piece of testcode—an assembler version of K. & R.'s immortal classic:

```
        IDNT   Hello_World

        XREF   Open
        XREF   Write
        XREF   Output

        SECTION text,CODE

START   jsr    Output        ; get current output stream
        tst.l  d0            ; test for success
        beq.s  quit          ; exit if error
        move.l d0,d1         ; get file handle
        move.l #Hello,d2     ; get message string
        moveq.l #size-Hello,d3 ; get length
        jsr    Write         ; call DOS
quit    rts

Hello   DC.B   'Hello World',$A,
size

        END
```

If we look at the disassembly of the code after it has been loaded by the DOS loader, it looks like:

```
21D9A8  jsr     $C04B94.l
21D9AE  tst.l   d0
21D9B0  beq.s   $21D9C6
21D9B2  move.l  d0,d1
21D9B4  move.l  #$21D9C8,d2
21D9BA  move.l  #$D,d3
21D9C0  jsr     $C04BA0.l
21D9C6  rts

21D9C8  48656C6C6F20576F726C640A0000  Hello World^J
```

and when the program was loaded, the DOS library node was listed as residing at `C04BD0`, so the addresses for the `jsr` instructions are correct! How does this load-time relocation work?

Method Used

Of the code fragments above, only the external definitions source was assembled with a Metacomco-compatible assembler. The other was assembled with Lattice's *ASM* because is it faster and a better assembler. The Lattice *ASM* program does not obey the rules; it generates non-standard `ext_hunks` to define absolute externals by prefixing the hunk with the name `_Abs` and setting the `EXT` type incorrectly to `0x01`, and hence cannot be used (I have nothing against Lattice—I'm a registered user of Lattice 5.02 and am very happy with it).

The original *Alink* was used to link the object code modules to produce the executable load file. Blink is not capable of generating `hunk_header` hunks in the load file with resident library references let alone cope with the undocumented hunk type `hunk_resident`, in the object code. I cannot comment on the suitability of Manx Aztec but I suspect it will be in the same league as Lattice.

The command lines to perform the assemble and link process are:

```
assem   dos.asm -o dos.o          ; the resident libraries
asm     hello.asm                 ; the "Hello World" code
mkres   dos.o resdos.o            ; transform the module
alink   resdos.o+hello.o TO Hello ; the final executable
```

What is Load-time Linking?

Load-time, or run-time linking of resident libraries is an extension to the already familiar load-time dynamic relocation of 32-bit values between code and data areas. If a program consists of multiple modules, code within one module may refer to code or data in another or even to locations within itself. Because the code can be scatter-loaded anywhere in memory by the DOS loader, the code cannot know before it is actually loaded, where the various parts of itself reside physically in the Amiga's memory. To this end, there is relocation information in the load file, identifying the locations that have to be "fixed up" to reflect the final address of the code.

Referring to Fig. 1, a diagrammatic simplification of the *reloc32* process, we have a location in hunk 1 referencing a byte location in hunk 2. During the linking process, the offset of that byte from the start of hunk 2 is calculated and stored in the relocation longword in hunk 1. It is noted that this longword location in hunk 1 will have to be relocated at load-time so an entry is made in hunk 1's relocation table. The relocation table specifies that the longword at our offset must be relocated with respect to hunk 2 when it is loaded. The work pays off at load time when the memory for the hunks is allocated, the loader adds the offset stored in hunk 1, to the base address of hunk 2, thus completely resolving the reference.

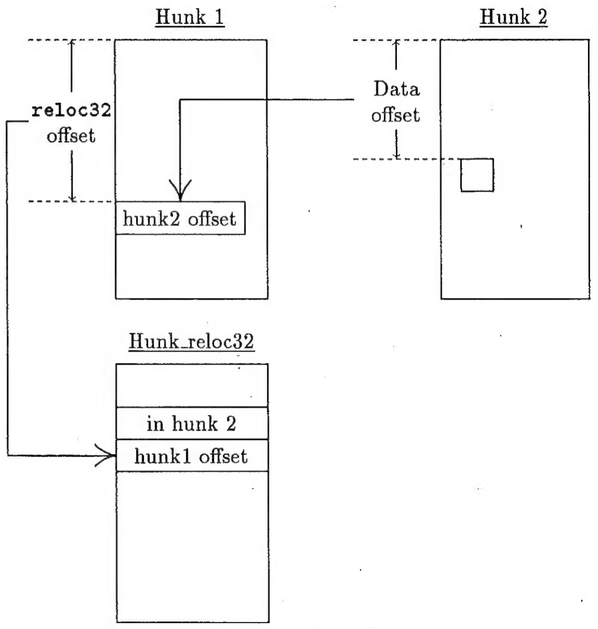


Fig. 1—The *reloc32* Process

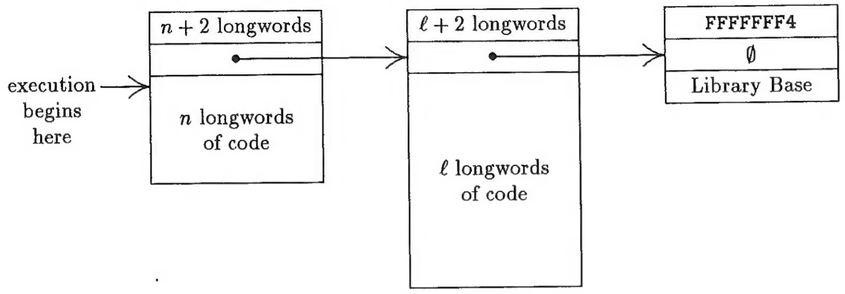


Fig. 2—The AmigaDOS *SegList*

The DOS Loader and Resident Libraries

When a piece of code is loaded by DOS, the loader allocates memory for each individual hunk in the load file of type `CODE`, `DATA` or `BSS`. These allocated regions of memory are called *segments* in AmigaDOS parlance. A BPTR (BCPL pointer) is returned from the DOS *LoadSeg*() function. This pointer (after conversion), points to a singly-linked list of segments which make up the program. Execution of the code is achieved by jumping subroutine to this *location* + 4 bytes.

With programs that use the resident library feature of the loader, the first hunk encountered in the load file, the `hunk_header`, contains the list of libraries in the order that they should be opened by the loader. These libraries are opened by a call to the Exec *OpenLibrary*() function, and when opened successfully, a special segment is queued to the partially completed segment list. This process occurs before loading any code so that time is not wasted if a library could not be opened. After loading, the *SegList* looks like the one in Fig. 2.

A resident library segment is a structure of 12 bytes with the base address of the library node structure as returned by *OpenLibrary*() at Segment address + 4, and the size of the segment (12 bytes), stored in one's-compliment form, at Segment-4 bytes.

Relocation of a longword in a hunk referring to a resident library occurs as if the library was just another hunk whose base address just happens to be the address of a library node. This is represented diagrammatically in Fig. 3.

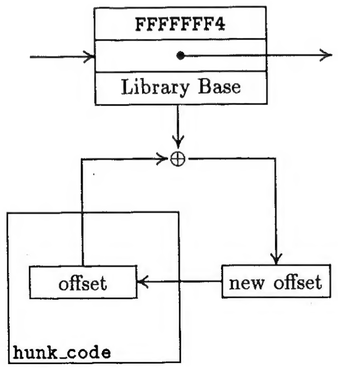


Fig. 3—Resident Library Relocation

Warning

The astute reader will notice something a little odd with "Hello World." We have come to expect all libraries to be called with their base address in register `A6`, yet in the example above, *DOSBase* is not loaded into any register. The DOS library is the exception to this rule; it is written in BCPL and the Exec-style front end to the library makes no demand that this convention be observed. This does not preclude the use of the resident library feature with other libraries as we can explicitly load `A6` with the library base using the same techniques as before and still call the routines by absolute reference:

```
movea    #IntuitionBase,A6
jsr      OpenWindow
```

How MKRES works

The *MKRES* program is the key to making the right resident library definitions available to the linker. *MKRES* converts the object code from the assembler into something that looks like:

- `hunk_unit`—name optional
- `hunk_name`—name of library in final load file
- `hunk_resident`
- *n*× `hunk_ext`
- `hunk_end`
- any number of the above group repeated for each library defined.

What does the final load file look like? Well, after all this, here is the output of the link phase of "Hello World:"

```
0000:  000003F3 00000003 646F732E 6C696272  ...?....dos.libr
0010:  61727900 00000000 00000001 00000001  ary.............
0020:  00000001 0000000B 000003E9 0000000B  ...........?....
0030:  4EB9FFFF FFC44A80 67102200 243C0000  N?????J.g.".$...
0040:  001C760D 4EB9FFFF FFD04E75 48656C6C  ..v.N?????NuHell
0050:  6F20576F 726C640A 00000000 000003EC  o World........?
0060:  00000001 00000001 0000000E 00000002  ................
0070:  00000000 00000002 00000016 00000000  ................
0080:  000003F2                             ...?
```

and we see a load file that probably hasn't been seen before by most Amiga programmers and possibly the shortest "Hello World" that can be written on the Amiga!

Well, I hope this has been interesting for you all; I bet you didn't know the Amiga could do that. It has been interesting for me as well—I love solving mysteries, and one of the mysteries of the Amiga that has vexed me for a while has finally fallen to a little detective work.

Don't think that this is the final word on the subject; There are several possibilities that I did not explore here—such as writing libraries to take greater advantage of this dynamic run-time linking process. A final thought on run-time linking—AmigaDOS had it long before OS/2; maybe we can re-discover its usefulness.

Connecting an NEC MultiSync-3D to the Amiga

by Eric Salter

THE Amiga's graphics output looks significantly more impressive on a monitor with good linearity and fine phosphor dot pitch. The NEC *MultiSync*-3D monitor is easily connected to an Amiga by anyone with a little skill with a soldering iron and pliers in about an hour. This article describes how to do it.

The *MultiSync*-3D comes with a standard 15-way VGA connector—ready to plug into an IBM but not very Amiga-friendly; the Amiga's RGB port is a 23-way 'D' connector which is not very friendly period! A simple solution is to the problem is to make a little adaptor plug using a 25-way 'D' connector shell.

The connectors should be wired as here:

VGA 15-Way		Amiga 23-Way	Signal
1	⟶	3	Analog Red
2	⟶	4	Analog Green
3	⟶	5	Analog Blue
4	•		
5	•		
6			Signal Ground
7			Signal Ground
8			Signal Ground
9	•		
10	•		
11	⟶	13	Signal Ground
12	•		
13	⟶	11	HSYNC
14	⟶	12	VSYNC
15	•		

In the table above, the '•' means that the pin is not connected and *Signal Ground*, is the ground return signal. Pins 6,7,8 and 11 of the VGA connector should be wired together and thence to pin 13 of the Amiga 23-way.

Both the female 15-way VGA and female 23-way 'D' connectors, can be obtained from *Radio Parts* in Spencer St. city. They are reasonably cheap and the service excellent.

The first step in creating the interface plug is to nibble the space for the VGA connector out of the 'D' shell. A nibbling tool is very useful but a rat-tail file is just as good as long as the shell is plastic. The hole is made in the narrow end of the shell—the other end is unmodified and fits the 23-way.

Once the hole is nibbled sufficiently, the metal tags on either side of the connector can be bent with pliers so they come to fit neatly inside the shell, securing and immobilizing the connector between the wall of the shell and the bit of plastic that holds the cable clamps.

Wired up, the 23-way and 15-way connectors fit neatly inside the shell. All that is left to do is bolt the thing together and plug in the *MultiSync*.

Mouse Balls!

Some of you may recognize the following article—it came from a Unix network. I'm told that it's a legitimate service note from IBM. You can make up your own mind...

—Perry Rosenboom

ESD PRODUCT SERVICE SUPPORT
SUBJECT:NEW RETAIN TIP

Record number:	H031944
Device:	D/T8550
Model:	M
Hit count:	UHC00000
Success count:	USC00000
Publication code:	PC50
Tip key:	025
Date created:	O89/02/14
Date last altered:	A89/02/15
Owning B.U.:	USA

Abstract:

MOUSE BALLS NOW AVAILABLE AS FRU (Field Replaceable Unit)

Text:

Mouse balls are now available as a FRU.

If a mouse fails to operate, or should perform erratically, it may be in need of ball replacement. Because of the delicate nature of this procedure, replacement of mouse balls should be attempted by trained personnel only.

Before ordering, determine type of mouse balls required by examining the underside of each mouse. Domestic balls will be larger and harder than foreign balls. Ball removal procedures will differ, depending upon manufacturer of the mouse. Foreign balls can be replaced using the pop-off method, and domestic balls replaced using the twist-off method. Mouse balls are not usually static sensitive, however, excessive handling can result in sudden discharge.

Upon completion of ball replacement, the mouse may be used immediately.

It is recommended that each servicer have a pair of balls for maintaining optimum customer satisfaction, and that any customer missing his balls should suspect local personnel of removing these necessary functional items.

P/N33F8462 – DOMESTIC MOUSE BALLS
P/N33F8461 – FOREIGN MOUSE BALLS

USERID (RSSTEWART) NODEID (BCRVM1)
INT.ZIP 1225, DEPT 2AW, TL 443-4597
(407-443-4597)
ESD PRODUCT SERVICE SUPPORT,
BOCA RATON, FL.

SkyLine Bulletin Board System Software Review

by Gary Gajic.

AN exciting new era has taken over the way bulletin boards (BBS') are being run on Amiga's. This new software call *SkyLine* has a few amazing features which will revolutionize future BBS software. Some of the features are as follows:

- IFF Brush Transmission (Automatic)
- Med-Res Graphics which include Lines, Boxes, Filled areas etc.
- Mouse Driven Menus, No more typo's.
- *X*, *Y*, and *Zmodem* File Transfer Protocols.

What You See at Your End

At the user's end is where you see all the good stuff. When logging onto *NWAUG ONE*, the logon screen is seen as 4 NWAUG balls with a bit of coloured text and some information about the system. After entering your name and password (the one that is always forgotten), there is an impressive advertisement for *Kev's Computer Shop* (Kev lent us the hard drive for a month or so). After that, the meeting times for AUG & NWAUG meetings and other information. Once at the main menu, it's like any other BBS' *files* section and *message* section. There also is a menu option for text libraries—that is where we can find all the special features of *Skyline* (More on that later). If you have never used *Skyline*, I know what your thinking... "I'll be waiting for hours for all this excellent graphics to be seen," but your wrong; *Skyline* is very ingenious in how it renders all its graphics. Rather than the old ANSI method of graphics where it started at the top left hand corner of the screen and ended at the bottom right, which was very VERY slow, all you see now are flashes and there are all the graphics!

How It Works

This may be complicated to some people, but I'll try and make it simple. Let's start with the simplest graphics item– a dot or PIXEL on the screen. For *Skyline* to show a dot on the online users screen it sends the following code:

```
ESC ]# X,Y,CL
```

where `ESC` is the *escape* key (`Chr$(27)`), '`#`,' is the type of graphic e.g., Dot, Line, Box etc., etc.; '`X`,' the x coordinate, '`Y`,' the y coordinate and '`CL`,' the colour of the graphic. So all *Skyline* has to send for a user to see a dot is 6 characters. This feature also applies for lines, boxes, filled boxes, circles and filled circles. If you have programmed in BASIC or 'C' (or most other languages for that matter), you can see a basic outline of how this feature works.

Another powerful feature is the *IFF brush* transferal of the system. The on-line user can also receive brushes which are basically parts of pictures, and have them displayed on the screen. As I mentioned earlier about the four NWAUG balls, the user only has to receive the brush once and then store it on a media for later use. The brush can also be stamped if you like as many times as needed. So rather than having to download the brush 10 times (Which by the way is automatically done via software), it only receives it once and then stamps it where needed.

Another feature is the *FONTS allocation* feature. A user can see any font transmitted by the BBS software e.g.,—on a stock *Workbench* disk there are seven fonts, and each font has about two different sizes. Lets use the *Diamond* 20 point font for an example. Now the BBS might send the following text—"The Cat Sat On The Mat"; it will the tell the comms program (At the users end) to load the *Diamond* 20 point font and use it to display the text—hence, you see the above text as a *Diamond* font.

You might think that the BBS software is very ingenious but it's not, the comms software (at the users end) is the smartest part of the software—it interprets all the control characters which are send and then uses them to display the graphics. This software was written especially for this type of BBS system by Michael Cox who thought up the whole *Skypix* protocol. Other programs such as *JRCOMM–99* also supports this protocol.

User Interface

This user interface is the simplest I have ever seen: Rather than using the keyboard for menu options, the on-line user can use His/Her mouse to select where they want to go e.g., *files* section, *message* section etc.

All in all it's a very complicated set up, but is so easy to use...

User's Problem

Here are a few problems that uses have had with *SkyTerm* 1.2 software:

When starting a download the computer asks for volume `TMP:` to be inserted into any drive. What is happening here is the computer wants to know where to put the downloaded file.

Solution—Assign `TMP:` to `RAM:` or to a floppy drive path. For more info consult Amiga DOS Manual or add the following to your Startup-Sequence:

For Floppy users

"`Assign TMP: RAM:`"—Downloaded file will go to `RAM:` or;
"`Assign TMP: DF0:`"—Downloaded file will go to `DF0:`

For Hard Drive users

"`Assign TMP: DH0:`"—Download file will go to `DH0:`

Another requester which may appear is the "`BRU:`" one. This is where *Skyterm* wants you to store the brushes which are downloaded for those amazing pictures I talked about earlier. Once again, a simple answer:

For Floppy users

"`Assign BRU: RAM:`"—Puts brush into `RAM:` or;
"`Assign BRU: DF0:`"—Puts brush onto `DF0:`

For Hard Drive Users

"`Assign BRU: DH0:Brush`"

The Last one is the best. Users can't get *Skyline* to dial. Once you have the *Phonebook* window up, you will see a modem string option which should show "ATDP," this is correct, now to enter a boards info, click with the left mouse button on the first entry (next to GO). Enter the BBS' name e.g., "NWAUG ONE," now click on the right hand side of the same line, this is where the number to be *dialed* goes.

For users who would like a complete working disk with *Skyterm* on it, it is available from the NWAUG PD Library.

In a nutshell, if you haven't seen this software in action, I suggest you get you hands on the software (Which by the way is Public Domain and is available on any Amiga BBS), give it a bash, you'll be impressed. Happy BBSing...

Sysop AmigaLink II, 376–6385, OPUS
Sysop NWAUG ONE, 376–7375, *SkyLine*

DME Update: Version 1.38

by Nikolai Kingsley

THIS version was released early November, 1989 (which means that there is probably a new version out by now), and is available on Fish Disk 284. This program was the first public domain program I got (yes, it is FREEWARE, not Shareware, and I'm not counting the absolutely ancient games that Maxwell distributed with the first few 500s they sold), and I'm still using it.

For those of you new to the Amiga who are wondering 'what the hell IS this *DME* thing that people rave about' (and, yes, *DME* users do tend to rave, just as *UEdit* users do), well, it's a file editor, that you can use to type up notes with, alter your `startup-sequence` with, create batch files and programs with, and such. Big Deal, you say? Commodore supplies you with an editor—'*Ed*'—you say? And *Ed* is only 19564 bytes long, where *DME* is 58720 bytes long, you say?

You can stop 'say'ing about there (I won't, though)... to say that *DME* is a bit more flexible than *Ed* would be like saying Stephen Hawkings is a bit smarter than Dermott Brereton. How flexible is *DME*? Well, you can map ANY of the keys to ANYTHING else. You can stuff people up by mapping the 'F' key to 'P' and vice-versa (as Benny Hill once did). Seriously, though, you can even map combinations of the mousebuttons and the keys, like:

```
map L-mmove (tomouse tlate 32)
```

which translates to 'whenever you press the left mousebutton, move the cursor to that spot and translate the character under it to a space,' which I find handy for erasing stuff.

Also, with version 1.38 (possibly with earlier versions, also—I have been a bit random about keeping up with updates) you can add a menu strip to the top, as follows:

```
menuadd File Load (newwindow arpload)
```

which will add a menu called `File` if there isn't one there already—if there is, it will add an item called `Load` to it, and when you select this menu item, it will perform the commands `newwindow` (open a new editing window—something else *Ed* can't do) and `arpload`, which will bring up the file-requester built in to *Arp* (of course you have *Arp* installed on your system!) to get the name of the file you want to load. *DME* in its original state doesn't have menus—if you press the left mouse button in that state, it 'iconifies' the window—shrinking it to a menu-bar-sized box in the top-left-hand part of the screen.

As you may have noticed, I've mentioned 'commands'. These commands, (and 120 others) are built into *DME*, and perform functions as simple as 'move the cursor right one space,' and as complex as CTAGS, which 'searches for the subroutine name under the cursor in the associated tags file ("tags" in the directory holding the file currently being edited)', which is probably more useful to *Aztec* 'C' programmers than to the ordinary Ivan who just wants to take the '`ENDCLI >NIL:`' out of his `startup-sequence`. These commands can be combined with an `if/while/else` command, *ARexx* macros (don't ask me—I just read the documentation), keys can be remapped on the fly (if you find yourself writing an essay about Khazakhistan, you can easily map the `F6` key to that string, which easily saves fifteen seconds if you are a three-finger typist like me), and, one of my favorite commands, '`SETFONT font sz`' which will let you use any size font (although be warned—proportional fonts do act rather strangely, as anyone who has used the CLI utility '*Setfont*' will know).

Something that rather annoyed me about the previous release of *DME* is that when you started it up, it would, by default, open a window in the middle of the screen, some sixty characters wide. I could fake it into opening a full screen window by putting the following into my shell-startup batch file:

```
Alias Ed run c:dme -t11 -l0 -w640 -h245 []
```

(well, you don't expect me to type all that in whenever I want to just look at a documentation file, do you?) Anyway, *DME* version 1.38 comes with a program called *DME-Config*, which asks you how big a window do you want, where do you want it, what *Workbench* colours do you want, u.s.w... then it modifies the *DME* program to conform to your specifications. Very nice. (I left most of the above '*alias*' command in the `shell-startup`, because it's handy to have *DME* running as a background task—it leaves the Shell free to do things like list directories and filenames, or see if there's room on `DF0:` for the novel you've just typed in).

Seasoned *DME* users know that when you start *DME*, it looks for a file called `.edrc` for default keymappings that you may want to use. This is my `S:.edrc`... (without the comments):

```
map s-f6 'findr '^m' ''
          /* removes carriage returns from files */
map a-f6 'findr '␣␣' '␣' left'
          /* converts two spaces to one. handy
          for packing 'justified' text files */
map L-mmove (tomouse tlate 32)
map alt-f10 (newwindow newfile dh0:system/dme.doc)
          /* on-line help! */
map A-f10 (tlate +1)
          /* changes an 'A' into a 'B'. handy for
          putting alternate characters in. */
map A-f9 (tlate -1)
```

```
            /* and taking them out again */
map a-n ((NoSan\'No\'Os))
            /* NoSan'No'Os—something I couldn't be
            bothered typing in again and again... */

/* my own menu commands. */
menuadd File Load (newwindow arpload)
menuadd File Insert (arpinsfile)
menuadd File Help (newwindow newfile
            dh0:system/dme.doc)
menuadd File (Hide) (Iconify)
```

With the first command, '`findr '^m' ''`'—this only removes the next carriage return. It would be simple to put this search-and-destroy command into an '`IFELSE`' loop, and get them all—I'm just lazy.

And, Norm, now it can be run from the Workbench.

The only complaints I have with this program (that's right, complaints yet) are: if you are in word-wrap mode, and you copy a line of text that is wider than the current window width, it hangs. Sometimes. Also, I have used a keymap editor to remap the '*help*' key to type '`endcli`' & (carriage return). If I press the '*help*' key, it types '`endcli`' into the document and gives a message '`Unknown Command - No Macros: ARexx Not Installed.`' I wonder what would happen if I installed *ARexx*?

gotta go now... "deepdown trauma hounds scratching at the door..."

—nikolai

The First Time User

by Eric Salter

SO, you've gone and bought yourself an Amiga have you? You race home from the store, rip open the packaging and find more cables and bits of paper than you can poke a stick at. The box is full of little messages on little cardboard cards saying—*CAUTION, do not attempt...* and *WARNING, please read the instructions before...*, or worse *Es ist verbotene...* If you bought yourself a 2000, you'll discover German all over again. After several hours, you figure out how to put it all together (and you thought your course in high Abyssinian Greek wouldn't come in handy) and you're ready to switch on.

Discovering that mains power makes all the difference, you plug it in and power up. Beautiful colour comes to a once matt-black screen and you are greeted with a hand holding a picture of a disk, asking you to insert *Workbench.* Upon reading the manual, you find *Workbench* is one of the disks you discovered lurking in the packing material, which is now scattered across most of the room and well into the next.

Inserting *Workbench* into the disk drive makes your Amiga emit strange grinding noises and you worry that the disk is being shredded. After some time, the *Workbench* screen appears in a fairly grotesque blue, orange, white and black combination but joyous day, your very own Amiga has sprung to life with windows, icons, mice and pull-down menus. You play with opening disk icons and moving windows, starting several clocks and covering the screen with them—oh boy, multitasking! Inquisitively, you click on the icon labeled '*Shell*' and a window appears with '`1>`' in it. You type at it and manage to obtain cryptic messages like '`Unknown command "vnsdkj".`' "How do I get rid of it?" you ask in muted tones—finally giving up and pushing it to the background with these really neat gadgets on the top of the window.

You play for a while but become tired of opening calculators and clocks and trying to close CLI windows. You look around you at the mess of paper, packing foam and manuals and think—"Is this all there is; is there not more to Amiga computing?"

In fact there is more to Amiga computing as you discover when your original *Workbench* disk, with which you had just been playing, refuses to be read—you then meet another aspect of Amiga life, the *system requester*, telling you the disk is un-readable and you should call the doctor. At this point, you haven't the foggiest what to do but such is the lot of the first-time user.

The scenario portrayed above is not uncommon. It has happened to us all at some time in our Amiga lifetime and it has a universal effect—panic, consternation and bewilderment. The documentation that arrives with a new Amiga, albeit better than the halcyon days of the A-1000, is still confusing—mainly because in our enthusiasm to use the blessed machine, we don't read it. When trouble strikes, and particularly with new users who are not "at home" with computers, it leaves a bad taste in their mouths and they are a little reticent to get back up on the horse that has thrown them and try again.

The Amiga is a great, innovative computer—don't get us wrong. If we had wanted 640 K of memory or monochrome, we would have bought an IBM or Mac, but we knew the Amiga was a cut above those—powerful, colourful graphics, high-fidelity sound and an operating system who's power makes the others pale into insignificance. With this power and flexibility comes a steeper learning curve than with other, more traditional computers, but when mastered, the Amiga outshines all others of similar price. Amiga really is the best kept secret in the computer industry.

The Amiga Users Group does not believe in keeping the secret. We would like to share with new and old users alike, our experiences with the Amiga. We want to support the new user through this difficult learning time so they can be doing useful, productive things with their new investment.

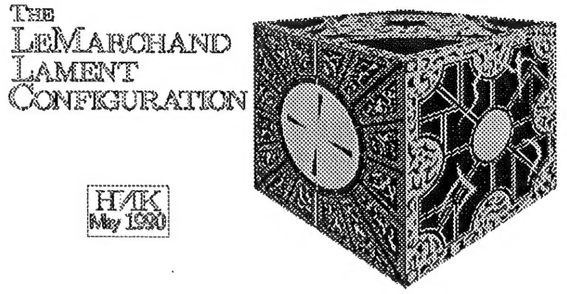
To this end, the group holds beginners classes for people who have just taken their machine out of the box. We explain how the Amiga should be set up, care of the original disks and finally, a guided tour through the *Workbench* interface, explaining how to use a mouse, open draws, run programs and importantly, make backup copies of original disks. May I encourage those who feel a bit bewildered by Amigas and computing to take up our offer and come along to our classes and take those first steps—they are an investment.

While you are waiting for one of these classes, let us share some general principles of computing that will stand you in good stead now and in the future.

- If in doubt, read the manual.
- If you don't know something—ask a friend or someone from the Amiga Users Group.

- Before doing anything on your new Amiga, make backup copies of the *Workbench*, *Extras* and if you have it the *Kickstart* disks.
- Always make backup copies of original disks and then put the originals away in a place free of dust and magnetic fields—only use the copies for day to day use.
- Keep the *write protect* tab set to open (write protected) unless you are actually going to write to or format the disk. When it is open (you can see through it) the little slider on the top right hand corner of the disk prevents the computer writing information to the disk. This may prevent accidental writes to the disk and corruption of your valuable data. It also prevents malicious writes to your disk by certain computer viruses.
- Never put a disk into your computer without checking it for a virus—this includes any programs you may download from bulletin boards or have given to you from friends. Never assume your friend is virus-free. Virus protection programs are available in the public domain software library of the Amiga Users Group.
- Protect your work by making regular (every half hour) backups of files. This saves you re-entering your data or re-writing your manuscript if the power fails after 4 or more hours of work.
- Do not copy or use commercial programs that you do not own i.e., DON'T PIRATE SOFTWARE. We all loose in the end because no one will write software if no one buys it.

Finally, enjoy your new machine and share your experience with others in the group. Remember, the Amiga Users Group is here to help.



AUG Adds

For Sale

Polaroid Palette for the Amiga. Complete with software driver, two cameras and all other standard accessories. Value over $4000, will sell $3200. Phone: (03) 783–9859.

—T. Michelin

Special Interest Groups

N.W.A.U.G.

THE North West Amiga Users Group is a sister group of the Amiga Users Group (AUG), to cater for Amiga users in the Northern and Western Suburbs. We are a dedicated group of Amiga enthusiasts who come together to explore the wonders of the Amiga computer

The Club has now been in operation since 1987 and has grown to more than 180 members strong. You don't have to be a brilliant computer programmer to join the club, just a passion for Amigas!

So what is it about the Amiga that has made it so popular? Probably the fact that no other home computer can offer as much as the Amiga. From incredible arcade quality games, to advanced simulations, to sophisticated CAD and animation, to superb music and sound capabilities, to innovative word processor and desk-top publishing applications, the Amiga covers all aspects of computing.

Why the Club then? Basically it's a meeting place so users with similar interests can come together and share the findings, ask questions, learn something new, and discover new ways of using their Amiga.

Members get to enjoy the benefits provided by the club, including a Public Domain Software Library, Magazine and Books Library, and access to Amiga Link II, a computerised Bulletin Board System via a modem. Discounts are also available from several retailers, including Kev's Computer Shop, Myer's, and the Technical Book Shop.

Membership costs only $7 a year. Your membership card entitles you to use the libraries and receive your discounts. $1 at the door for each meeting you attend and that's all there is to it.

Meetings are held at the *Essendon Community Centre*, corner of Pascoe-Vale and Mount Alexander Roads in Moonee Ponds every Wednesday fortnight, starting on 1*st* August at 7:30 p.m. will about 10:30 p.m.

So come along and be part of the action! Hope to see you there...

—Paul Pritsis (NWAUG Secretary)

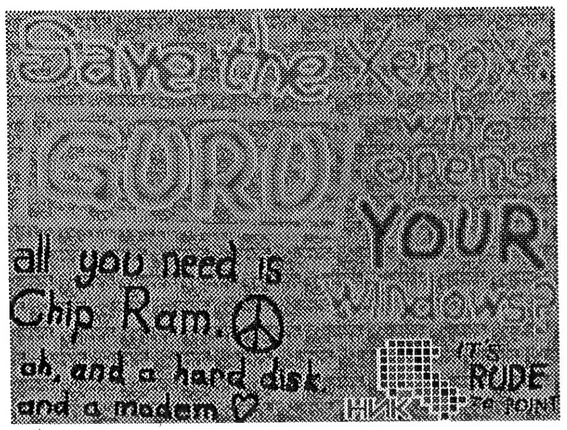
NWAUG Committee

Co-ordinator	Hugh Leslie	489 1584	Clifton Hill
Ass. Co-ord	Kid Currie	531 7282	Elsternwick
Meeting Chair	George Wahr	376 6180	Kensington
Mem. & Treas.	Lawrence Hill	478 6145	N/A
Pub. Domain	Alan Cheng	380 5588	West Brunswick

Amiga Help Network

The following is a list of AUG members who have volunteered to share their knowledge/experiences with others. If you also want to help and have your name listed here, please contact Lester McClure (233–5664 A.H.). The names are not listed in any order of priority and the format may change in future listings. Please keep contacts to reasonable hours (6–9 p.m. unless otherwise mentioned) and remember one very important basis of this service—they are volunteers...

Name	Area of Expertese	Phone #
Neville Sleep	AmigaBASIC (Beginner level)	546 0633
Rudy Hohut	AmigaBASIC (Intermediate)	807 3911
John Elston	AmigaBASIC (Advanced)	375 4142
Alan Garner	AmigaBASIC, A/C BASIC	879 2683
Mal Woods	'C' (Introductory), Professional Page	888 8129
Andrew Gelme	'C' (Advanced)—AZTEC	645 1744
Eric Salter	'C' (Advanced)—Lattice, TEX	853 9117
Norm Christian	Amiga art, music	798 6552
Neil Rutledge	Music, audio sampling, MIDI	578 5724
Russ Lorback	Excellence !, Superbase Professional (After 9:30 p.m.)	756 6640
Darren King	Amiga viruses, modems/communications	546 5040
George Wahr	Side-car, Bridgeboard	376 6180
James Gardiner	AmigaDOS, auto-boot hard drives	532 8030
Lester McClure	Lucas/Frances—A1000 32-bit processor system	233 5664
Joe Santamaria	Graphic arts, DPaint, Sculpt *etc.*...	836 9129



The Editor's Desk

AS you can see, the flavor of this month's newsletter is very different. It was just an experiment to see just how TEX handled the format and to test my skill as a TEX programmer. I hope you have enjoyed the journal as much as I did while I was coercing TEX to perform typographical gymnastics.

Unfortunately, I was unable to show off the *encapsulated PostScript* inclusion of AmigaTEX in the article—'Connecting an NEC MultiSync...'; I did have some nice diagrams to include but I encountered TEX in one of its bad moods!

I was also thwarted by a distinct lack of articles. Apart from our faithful and prolific Nikolai Kingsley *et al.*, there was NO COPY! Instead, you got a lot of me, but what I call 'interesting subjects,' are probably not what you want to read about. Nonetheless, you got some quality just the same.

Next month, you will have Con taking the con (sorry) again, and he will be needing lots of articles. May I suggest a concerted effort. You all have something to contribute to the Amiga community—the Amiga is far too big for any one individual to come to grips with it by themselves. This journal is the ideal way to tell others what you are doing with your Amiga.

Anyway, It has been a fun time—let me know if you'd like to see more of what TEX can do. Best wishes

—Eric Salter (Guest Editor)

Write for Workbench
Fame, Fortune etc.
Free P.D. Disks!

"Don't you try to outweird me, I get stranger things than you free in my breakfast cereal"

—Zaphod Beeblebrox

PUBLIC DOMAIN SOFTWARE ORDER FORM

Mail to: Amiga Users Group, PO Box 48, Boronia 3155, Victoria

Disk Numbers:										

Don't forget to specify collection name i.e., Fish, Amigan, Amicus etc.

Disks supplied by Amiga Users Group @ $6 each	$
Disks supplied by member @ $2 each	$
Club Use Only:	Total: $
Member's Name:	Membership #:
Address:	
	Postcode:

NEWSLETTER BACK ISSUE ORDER FORM

Mail to: Amiga Users Group, PO Box 48, Boronia 3155, Victoria

Issue Numbers:										

Be patient, we may have to reprint some issues to fill your request

Number of issues ordered @ $2 each	$
Club Use Only:	Total: $
Member's Name:	Membership #:
Address:	
	Postcode:

APPLICATION FOR MEMBERSHIP OF THE AMIGA USERS GROUP INC.

Membership is $25 per year. Send your cheque to: Amiga Users Group Inc., PO Box 48, Boronia, 3155

Surname: _____

First Name: _____

Address: _____

_____ Postcode: _____

Phone Number: _____ STD Code: _____

Where did you here about AUG: _____

_____

_____

Signed: _____ Date: _____

Details on this side are optional

Year of birth: _____ Which Model Amiga _____

Occupation: _____

Interests: _____

_____

_____

Dealer's Name: _____

Dealer's Address: _____

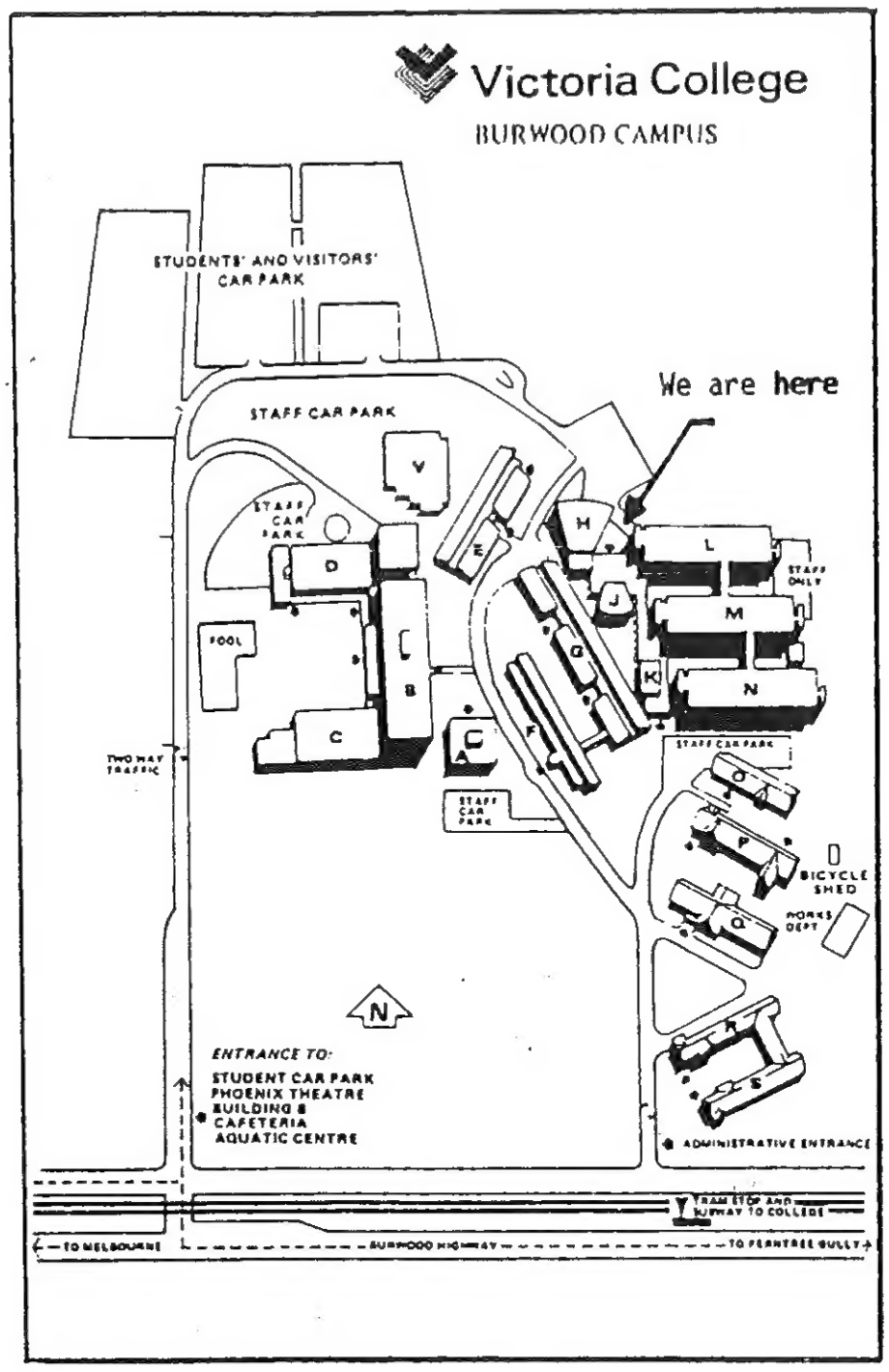
_____

If admitted as a member, I agree to abide by the rules of the Association for the time being in force

Club Use Only	Date	Paid	Rcpt #	Memb #	Card Sent

August 1990 Amiga Workbench

AUG normally meets on the third Sunday of each month



Where is Victoria College Burwood Campus?

Melways Map 61, reference B5

People often have difficulty locating our meeting place the first few times. Victoria College is on the North side of Burwood Highway Burwood, just East of Elgar road. Coming from the city along Burwood Highway, turn left at the first set of traffic lights after Elgar road. Follow the road around past the football oval, over five traffic bumps, to the car parking area near the netball courts. Further up the road, to the right, you'll find Lecture Theatre 2.